AMENDMENTS TO THE CLAIMS

1. (Currently amended) A method of translating a subject code executable by a subject computing architecture into a target code executable by a second computing architecture, wherein the subject code includes at least a first program and a second program, comprising the steps of:

providing a first translator instance which translates the subject code of the first program into the target code including translating a first portion of the subject code into a portion of the target code;

caching said portion of the target code into a shared code cache facility; ~~and~~

providing a second translator instance which is different from the first translator instance and which translates the subject code of the second program into the target code, wherein the second translator instance operates simultaneously with the first translator instance;

~~including~~ retrieving the cached portion of the target code from the shared code cache facility upon a compatibility detection between said cached portion of the target code and a second portion of the subject code in the second program, including loading the portion of the target code in the shared code cache facility into a portion of memory which is shared amongst at least the first and second translator instances; and,

copying at least one part of the shared code cache facility to a private portion of memory associated with the second translator instance upon modification of the at least one part of the shared code cache facility by the second translator instance.

2. (Previously presented) The method of claim 1 wherein said compatibility detection is determined by a cache key comparison between a cache key associated with the first portion of the subject code and the second portion of the subject code.

3. (Previously presented) The method of claim 2 wherein the cache key is a byte sequence that encodes a subject code instruction sequence of the respective first and second portions of subject code.

4. (Previously presented) The method of claim 2 wherein the cache key is a hash of the corresponding a subject code instruction sequence of the respective first and second portions of subject code.

5. (Previously presented) The method of claim 2 wherein the cache key comprises: (1) a filename of an executable file containing the subject code; (2) an offset and a length of the subject code instruction sequence of the respective first and second portions of the subject code; (3) a last modification time of the executable file; (4) a version number of the respective first and second translator instance; and (5) a subject memory address where of the subject code instruction sequence was loaded.

6. (Original) The method of claim 2 wherein the cache key comprises a plurality of metrics.

7. (Previously presented) The method of claim 2 wherein the compatibility detection is determined by computing a cache key data structure corresponding to the subject code to be translated to a plurality of second data structures, each second data structure corresponding to a different set of cached target code instructions.

8. (Original) The method of claim 1 further including the step of executing the target code.

9. (Original) The method of claim 1 wherein translations of self-modifying code are not cached.

10. (Original) The method of claim 1 wherein the portion of target code cached comprises a translation structure including a basic block.

11. (Original) The method of claim 1 wherein the portion of target code cached comprises one or more block translations and their respective successor lists.

12. (Original) The method of claim 1 wherein the portion of target code is converted into a single cache unit comprising a subject program and all its associated libraries.

13. (Original) The method of claim 1 wherein the portion of target code cached consists of a single instruction.

14. (Original) The method of claim 1 wherein the portion of target code cached comprises all code blocks corresponding to the same starting subject address.

15. (Original) The method of claim 1 wherein the portion of target code cached comprises a cache unit representing a discrete range of subject addresses.

16. (Original) The method of claim 1 wherein the portion of target code cached as a unit comprises a subject library.

17. (Currently amended) A computer system, comprising ~~In combination~~:

a target processor;

a shared code cache facility; and

translator code for translating a subject program code into target code executable on said target processor, said translator code comprising code executable by said target processor to:

~~translate~~ provide a first translator instance which translates the subject code of a first program into the target code including translating a first portion of the subject code into a portion of the target code;

cache said portion of the target code in the shared code cache facility; and

~~retrieve~~ provide a second translator instance which translates the subject code of a second program into the target code, wherein the second translator instance is different from the first translator instance and operates simultaneously with the first translator instance, and wherein the second translator instance retrieves ~~including retrieving~~ the cached portion of the target code from the shared code cache facility upon a compatibility detection between said cached portion of the target code and a second portion of the subject code in the second program, including loading the portion of the target code in the shared code cache facility into a portion of memory which is shared amongst at least the first and second translator instances; and,

copying at least one part of the shared code cache facility to a private portion of memory associated with the second translator instance upon modification of the at least one part of the shared code cache facility by the second translator instance.

18. (Previously presented) The computer system combination of claim 17 wherein said compatibility detection of cache translations and subject code to be translated is determined by a cache key comparison between a cache key associated with the first portion of the subject code and with the second portion of the subject code.

19. (Previously presented) The computer system combination of claim 18 wherein a the byte sequence that encodes the corresponding a subject code instruction sequence of the respective first and second portions of the subject code.

20. (Previously presented) The computer system combination of claim 18 wherein the cache key is a hash of the corresponding a subject code instruction sequence of the respective first and second portions of the subject code.

21. (Previously presented) The computer system combination of claim 18 wherein the cache key comprises: (1) a filename of an executable file containing the subject code; (2) an offset and a length of the subject code instruction sequence of the respective first and second portions of the subject code; (3) a last modification time of the executable file; (4) a version number of the respective first and second translator instance; and (5) a subject memory address where of the subject code instruction sequence was loaded.

22. (Previously presented) The computer system combination of claim 18 wherein the cache key comprises a plurality of metrics.

23. (Previously presented) The computer system combination of claim 18 wherein the compatibility detection is determined by comparing a cache key data structure corresponding to the subject code to be translated to a plurality of second data structures, each second data structure corresponding to a different set of cached target code instructions.

24. (Previously presented) The computer system combination of claim 17 further including the step of executing the target code.

25. (Previously presented) The computer system combination of claim 17 wherein translations of self-modifying code are not cached.

26. (Previously presented) The computer system combination of claim 17 wherein the portion of target code cached comprises a translation structure including a basic block.

27. (Previously presented) The computer system combination of claim 17 wherein the portion of target code cached comprises one or more block translations and their respective successor lists.

28. (Previously presented) The computer system combination of claim 17 wherein the portion of target code is converted into a single cache unit comprising a subject program and all its associated libraries.

29. (Previously presented) The computer system combination of claim 17 wherein the portion of target code cached consists of a single instruction.

30. (Previously presented) The computer system combination of claim 17 wherein the portion of target code cached comprises all code blocks corresponding to the same starting subject address.

31. (Previously presented) The computer system combination of claim 17 wherein the portion of target code cached comprises a cache unit representing a discrete range of subject addresses.

32 (Previously presented) The computer system combination of claim 17 wherein the portion of target code cached as a unit comprises a subject library.

33. (Currently amended) A program storage medium storing translator code for translating subject ~~program~~ code into target code, wherein the subject code includes at least a first program and a second program, said translator code, when executed by a computer, being operable to perform the steps comprising:

providing a first translator instance executing on the computer which translates the subject code of the first program into the target code including translating a first portion of the subject code into a portion of the target code;

caching said portion of the target code into a shared code cache facility; and

providing a second translator instance executing on the computer, wherein the second translator instance is different from the first translator instance and wherein the second translator instance ~~which~~ translates the subject code of the second program into the target code, including retrieving the cached portion of the target code from the shared code cache facility upon a compatibility detection between said cached portion of the target code and a second portion of the subject code in the second program, including loading the portion of the target code in the shared code cache facility into a portion of memory which is shared amongst at least the first and second translator instances; and,

copying at least one part of the shared code cache facility to a private portion of memory associated with the second translator instance upon modification of the at least one part of the shared code cache facility by the second translator instance.

34. (Previously presented) The storage medium of claim 33 wherein said compatibility detection of cache translations and subject code to be translated is determined by a cache key comparison between a cache key associated with the first portion of the subject code and the second portion of the subject code.

35. (Previously presented) The storage medium of claim 34 wherein the cache key is a the byte sequence that encodes the corresponding a subject code instruction sequence of the respective first and second portions of subject code.

36. (Previously presented) The storage medium of claim 34 wherein the cache key is a hash of the corresponding a subject code instruction sequence of the respective first and second portions of subject code.

37. (Previously presented) The storage medium of claim 34 wherein the cache key comprises: (1) a filename of an executable file containing the subject code; (2) an offset and a length of the subject code instruction sequence of the respective first and second portions of the subject code; (3) a last modification time of the executable file; (4) a version number of the respective first and second translator instance; and (5) a subject memory address where of the subject code instruction sequence was loaded.

38. (Original) The storage medium of claim 34 wherein the cache key comprises a plurality of metrics.

39. (Previously presented) The storage medium of claim 34 wherein the compatibility detection is determined by computing a cache key data structure corresponding to the subject code to be translated to a plurality of second data structures, each second data structure corresponding to a different set of cached target code instructions.

40. (Original) The storage medium of claim 33 further including the step of executing the target code.

41. (Original) The storage medium of claim 33 wherein translations of self-modifying code are not cached.

42. (Original) The storage medium of claim 33 wherein the portion of target code cached comprises a translation structure including a basic block.

43. (Original) The storage medium of claim 33 wherein the portion of target code cached comprises one or more block translations and their respective successor lists.

44. (Original) The storage medium of claim 33 wherein the portion of target code is converted into a single cache unit comprising a subject program and all its associated libraries.

45. (Original) The storage medium of claim 33 wherein the portion of target code cached consists of a single instruction.

46. (Original) The storage medium of claim 33 wherein the portion of target code cached comprises all code blocks corresponding to the same starting subject address.

47. (Original) The storage medium of claim 33 wherein the portion of target code cached comprises a cache unit representing a discrete range of subject addresses.

48. (Original) The storage medium of claim 33 wherein the portion of target code cached as a unit comprises a subject library.

49. (Currently amended) A program storage medium storing translator code for translating subject code into target code when said translator code is executed by a computer, the translator code comprising: ~~In combination:~~

program code providing a first translator instance for translating a first portion of subject code in a first program into a portion of target code; and

program code for caching said portion of target code into a shared code cache facility and for providing a second translator instance for retrieving said target code from said shared code cache facility upon detection of compatibility between a second portion of subject code in a second program and said portion of target code, wherein said second translator instance operates simultaneously with the first translator instance, and

program code for loading the portion of the target code in the shared code cache facility into a portion of memory which is shared amongst at least the first and second translator instances; and,

copying at least one part of the shared code cache facility to a private portion of memory associated with the second translator instance upon modification of the at least one part of the shared code cache facility by the second translator instance.

50. (Cancelled)

51. (Original) The method of claim 50 wherein said target code is cached at the end of translation of said first program.

52. (Previously presented) The method of claim 50, wherein the first translator instance translates the first subject program including the first portion of the subject code into the portion of

the target code, and the second translator instance translates the second subject program including reusing the portion of the target code created by the first translator instance.

53. (Previously presented) The method of claim 1, further comprising the steps of:

copying the portion of target code from a private storage associated with the first translator instance to a shared code cache facility;

retrieving the portion of target code from the shared code cache facility for reuse by the second translator instance.

54. (Previously presented) The method of claim 53, further comprising the steps of:

selectively identifying one or more static target code portions amongst a plurality of portions of the target code produced by the first translator instance, wherein the static target code portions are derived from static subject code portions in the subject code; and

caching the identified static target code portions.

55. (Previously presented) The method of claim 54, further comprising the steps of:

identifying one or more dynamic target code portions amongst a plurality of portions of the target code produced by the first translator instance, wherein the dynamic target code portions are derived from dynamically generated portions of the subject code; and

discarding the dynamic target code portions.

56. (Previously presented) The method of claim 55, further comprising performing the steps of selectively identifying, caching, identifying and discarding upon completing execution of the first translator instance.

57. (Previously presented) The method of claim 1, further comprising:

providing a shared code cache facility to cache the portion of the target code; and

selectively replacing the cached portion of target code in the shared code cache facility where the second translator instance provides an updated translation of the portion of target code.

58. (Previously presented) The method of claim 57, further comprising the steps of:

publishing the portion of target code from the first translator instance to the shared code cache facility during execution of the first translator instance; and

retrieving the portion of the target code from the shared code cache facility for reuse by the second translator instance;

wherein the first translator instance and the second translator instance execute concurrently.

59. (Previously presented) The method of claim 58, wherein the publishing step comprises publishing at cache synchronisation points having a predetermined trigger condition.

60. (Previously presented) The method of claim 59, wherein the predetermined trigger condition is any one or more of:

(a) idle periods when the first translator instance is inactive;

(b) after a threshold number of translation structures have been generated; and

(c) upon a request by the second translator instance.

61. (Previously presented) The method of claim 1, further comprising the steps of:

providing a shared code cache facility to cache the portion of the target code; and

selectively performing optimisations of the shared code cache facility.

62. (Previously presented) The method of claim 61 wherein the optimisations comprise any one or more of:

(a) restructuring a cache directory structure of the shared code cache facility to make searches for a particular portion of target code more efficient;

(b) deleting translations that have been superseded by subsequent, more optimized translations of the same subject code;

(c) rearranging the shared code cache facility to locate frequently requested portions of target code near each other;

(d) performing offline optimizations of cached translations; and

(e) performing offline predictive translation to translate subject code which has not yet been translated by a translator instance but which a translator instance is expected to encounter.

63. (Cancelled)

64. (Previously presented) The method of claim 1, further comprising the steps of:

providing a shared code cache facility to cache the portion of target code; and

distributing the shared code cache facility amongst two or more caches.

65. (Previously presented) The method of claim 64, wherein the distributing step comprises providing scoped caches, ranged caches, or cache policies.

66. (Previously presented) The method of claim 1, further comprising the steps of:

aggressively optimising the translation in the first translator instance to provide an optimised portion of target code; and

reusing the optimised portion of target code in the second translator instance.

67. (Previously presented) The method of claim 66, further comprising the step of:

performing a less aggressive optimised translation in the second translator instance when translating the second portion of subject code for which a portion of target code is not cached.